

EXHIBIT E

IMPLICIT, LLC'S INFRINGEMENT ANALYSIS**U.S. Patent No. 8,056,075 – Capital One Financial Corp., Capital One, National Association****Claim 1**

Implicit, LLC (“Implicit”) provides evidence of infringement of claim 1 of U.S. Patent No. 8,056,075 (hereinafter “the ’075 patent”) by Capital One Financial Corp. and Capital One, National Association (collectively, “Capital One”). In support thereof, Implicit provides the following claim charts.

“Accused Instrumentalities” as used herein refers to at least the components involved in the dynamic generation of applications/applets/apps, including but not limited to webpages and mobile apps (or components thereof), which are dynamically generated by one or more Capital One servers running Node.js and V8 software in response to a request from a client device (e.g., a browser or mobile device used by a Capital One customer/subscriber/user to access data/information stored by Capital One) to the one or more Capital One servers. An example of the Accused Instrumentalities are the one or more servers that generate a response to a web-browser request directed to a Capital One webpage (e.g., <https://www.capitalone.com/>). These claim charts demonstrate Capital One’s infringement by comparing each element of the asserted claims to corresponding components, aspects, and/or features of the Accused Instrumentalities. These claim charts are not intended to constitute an expert report on infringement. These claim charts include information provided by way of example, and not by way of limitation.

The analysis set forth below is based only upon information from publicly available resources regarding the Accused Instrumentalities, as Capital One has not yet provided any non-public information. An analysis of Capital One’s (or other third parties’) technical documentation and/or software source code may assist in fully identifying all infringing features and functionality. Accordingly, Implicit reserves the right to supplement this infringement analysis once such information is made available to Implicit. Furthermore, Implicit reserves the right to revise this infringement analysis, as appropriate, upon issuance of a court order construing any terms recited in the asserted claims.

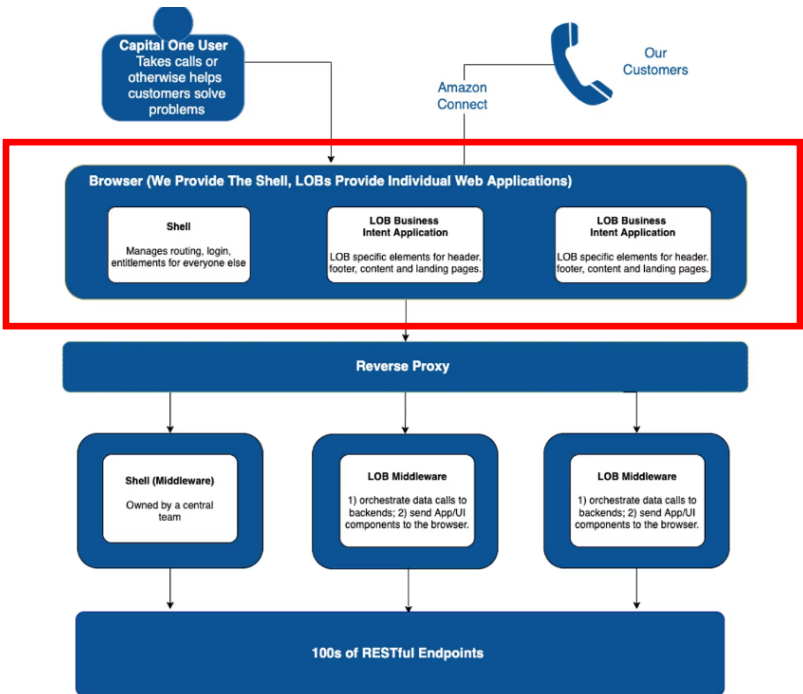
Implicit provides this evidence of infringement and related analysis without the benefit of claim construction or expert reports or discovery. Implicit reserves the right to supplement, amend or otherwise modify this analysis and/or evidence based on any such claim construction or expert reports or discovery.

Unless otherwise noted, Implicit contends that Capital One directly infringes the ’075 patent in violation of 35 U.S.C. § 271(a) by selling, offering to sell, making, using, and/or importing the Accused Instrumentalities. The following exemplary analysis demonstrates that infringement.

Unless otherwise noted, Implicit believes and contends that each element of each claim asserted herein is literally met through Capital One’s provision of the Accused Instrumentalities. However, to the extent that Capital One attempts to allege that any asserted claim element is not literally met, Implicit believes and contends that such elements are met under the doctrine of equivalents. More specifically, in its investigation and analysis of the Accused Instrumentalities, Implicit did not identify any substantial differences between the elements of the patent claims and the corresponding features of the Accused Instrumentalities, as set forth herein. In each instance, the identified feature of the Accused Instrumentalities performs at least substantially the same function in substantially the same way to achieve substantially the same result as the corresponding claim element.

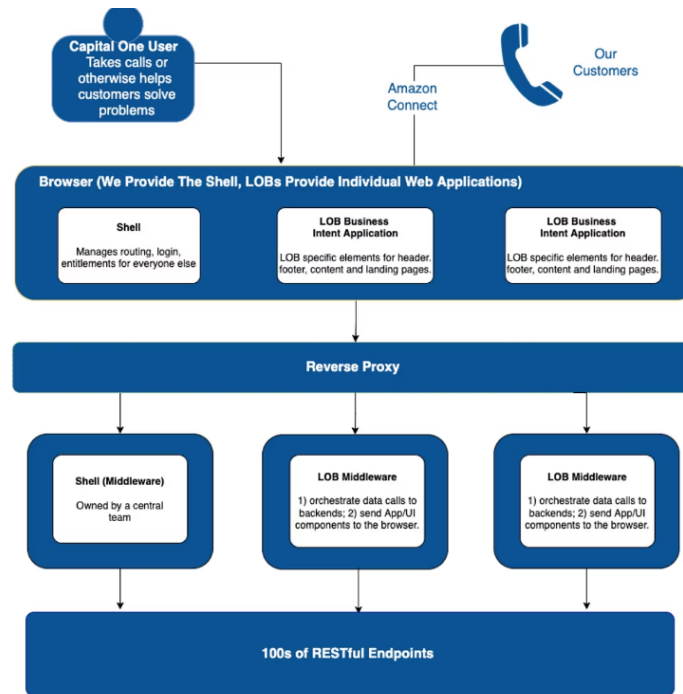
IMPLICIT LLC'S ANALYSIS OF INFRINGEMENT

To the extent the chart of an asserted claim relies on evidence about certain specifically-identified Accused Instrumentalities, Implicit asserts that, on information and belief, any similarly-functioning instrumentalities also infringe the charted claim. Implicit reserves the right to amend this infringement analysis based on other products made, used, sold, imported, or offered for sale by Capital One. Implicit also reserves the right to amend this infringement analysis by citing other claims of the '075 patent, not listed in the claim chart, that are infringed by the Accused Instrumentalities. Implicit further reserves the right to amend this infringement analysis by adding, subtracting, or otherwise modifying content in the "Accused Instrumentalities" column of each chart.

| Claim # | Accused Instrumentalities |
|--|---|
| <p>1. A method for delivering one or more applets to one or more client computers, comprising, in no particular order, the steps of:</p> | <p>The Accused Instrumentalities perform a method for delivering one or more applets to one or more client computers. This is illustrated, for example, as follows where the applet is https://www.capitalone.com/:</p>  <p>The diagram illustrates the front-end architecture. At the top, 'Capital One User' (with a person icon) and 'Our Customers' (with a phone icon) are connected to 'Amazon Connect'. Both point to a 'Browser' box, which is highlighted with a red border and labeled 'client' in red text. The 'Browser' box contains three sub-components: 'Shell' (Managing routing, login, entitlements for everyone else), 'LOB Business Intent Application' (LOB specific elements for header, footer, content and landing pages), and another 'LOB Business Intent Application' (LOB specific elements for header, footer, content and landing pages). Below the browser is a 'Reverse Proxy' box. Arrows from the browser point to the reverse proxy, which then points to three 'Middleware' boxes: 'Shell (Middleware)' (Owned by a central team), 'LOB Middleware' (1) orchestrate data calls to backends; 2) send App/UI components to the browser, and another 'LOB Middleware' (1) orchestrate data calls to backends; 2) send App/UI components to the browser. All three middleware boxes point to a final box labeled '100s of RESTful Endpoints'.</p> <p>Our front-end architecture matches pretty well to what Michael Geers might characterize as an “app shell” with multi-level routing. At the foundational level, we lean heavily on standard open source software libraries and practices. For backend libraries, we use (among others) Fastify, NestJS, Pino, Restify and Undici. On the front-end, we utilize Vue.js and React.js.</p> <p>All of our browser to backend connections go through Node microservices. These microservices are responsible for most of the middle-tier work you would expect in a distributed application. This includes data orchestration, business rule enforcement, and logical decisioning.</p> <p>Source: https://www.capitalone.com/tech/software-engineering/loosely-coupled-micro-frontends-with-nodejs/</p> |

configuring an applet server manager at a server computer to manage at least one request from the one or more client computers for the one or more applets, the applet server manager having access to one or more networks;

The Accused Instrumentalities perform a method that includes, configuring an applet server manager at a server computer to manage at least one request from the one or more client computers for the one or more applets, the applet server manager having access to one or more networks. This is illustrated, for example, as follows:

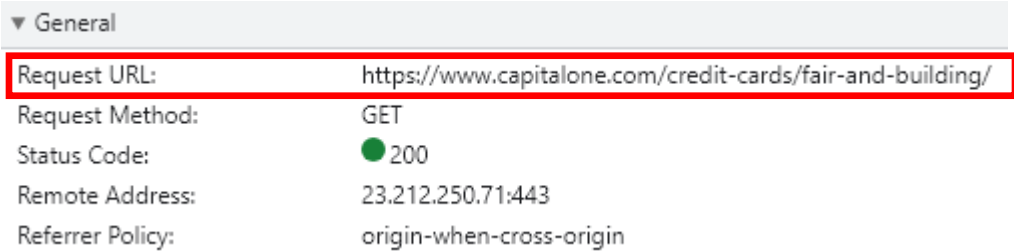


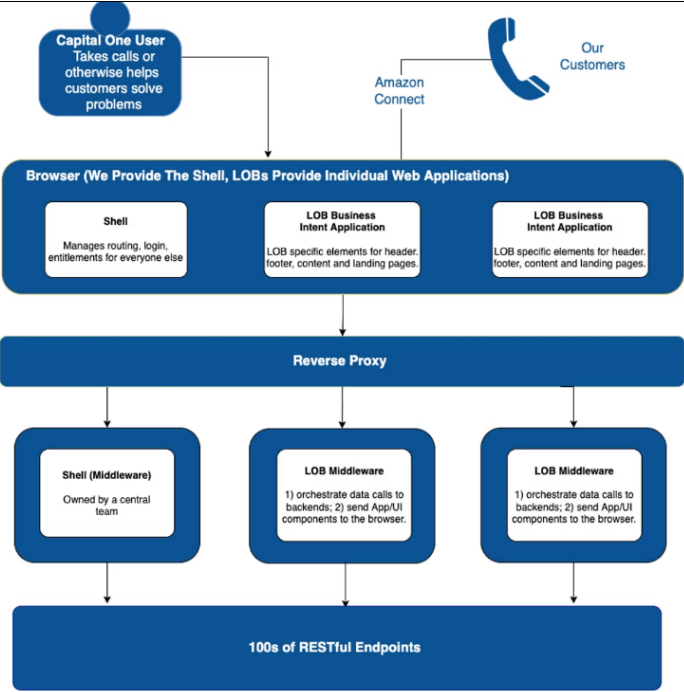
Our front-end architecture matches pretty well to what [Michael Geers](#) might characterize as an “app shell” with multi-level routing. At the foundational level, we lean heavily on standard open source software libraries and practices. For backend libraries, we use (among others) [Fastify](#), [NestJS](#), [Pino](#), [Restify](#) and [Undici](#). On the front-end, we utilize [Vue.js](#) and [React.js](#).

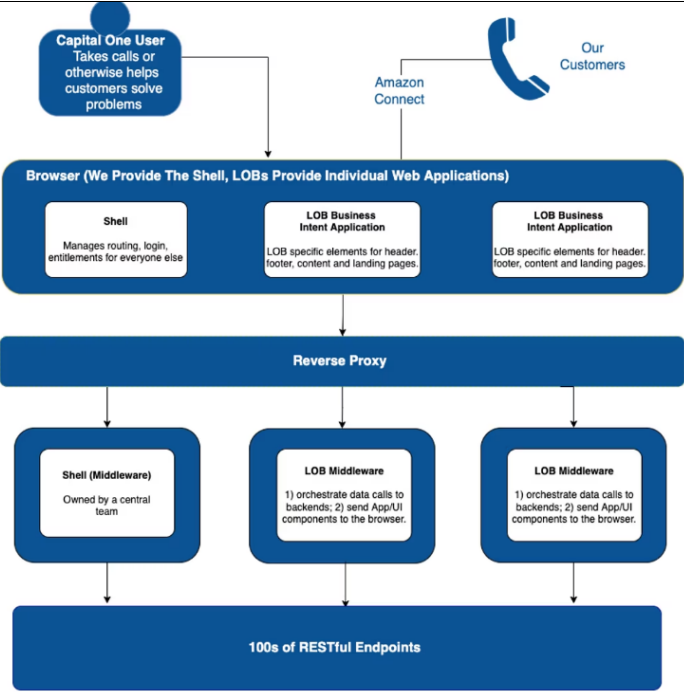
All of our browser to backend connections go through Node microservices. These microservices are responsible for most of the middle-tier work you would expect in a distributed application. This includes data orchestration, business rule enforcement, and logical decisioning.

server (backend)

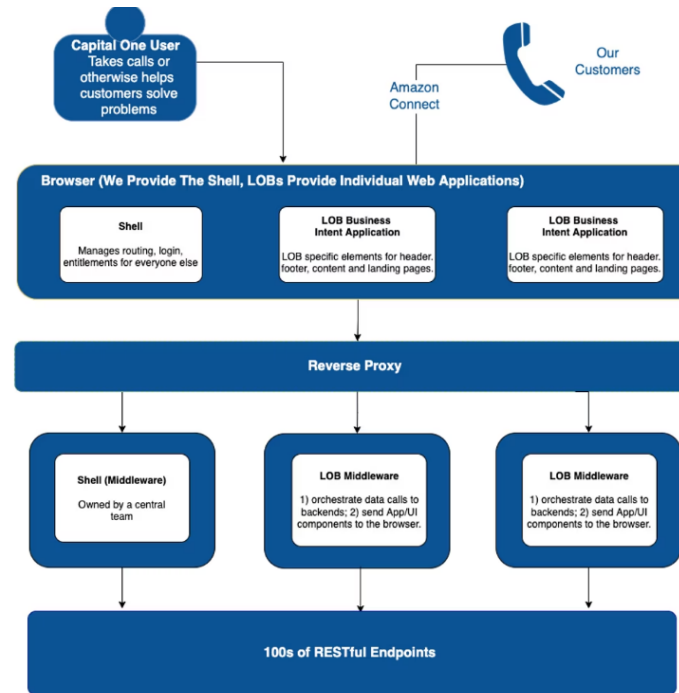
Source: <https://www.capitalone.com/tech/software-engineering/loosely-coupled-micro-frontends-with-nodejs/>

| | |
|--|---|
| | Upon information and belief, the applet server manager has access to one or more networks and manages at least one request from one or more client computers for the one or more applets. |
| receiving the at least one request at the applet server manager, | <p>The Accused Instrumentalities perform the step of receiving the at least one request at the applet server manager. This is illustrated, for example, as follows:</p>  <p>The screenshot shows a 'General' tab in a developer tool. A red box highlights the 'Request URL' field, which contains the text 'https://www.capitalone.com/credit-cards/fair-and-building/'. Below this, other fields are visible: 'Request Method' is 'GET', 'Status Code' is '200' with a green circle icon, 'Remote Address' is '23.212.250.71:443', and 'Referrer Policy' is 'origin-when-cross-origin'.</p> <p>Source: https://www.capitalone.com/credit-cards/fair-and-building/</p> |
| passing the at least one request from the applet server manager to at least one of the one or more networks; | The Accused Instrumentalities perform the step of passing the at least one request from the applet server manager to at least one of the one or more networks. This is illustrated, for example, as follows: |

| | |
|---|--|
| |  <p>The diagram illustrates the front-end architecture. At the top, 'Capital One User' (who takes calls or helps solve problems) and 'Our Customers' (via 'Amazon Connect') interact with the 'Browser (We Provide The Shell, LOBs Provide Individual Web Applications)'. The browser contains a 'Shell' (managing routing, login, etc.) and two 'LOB Business Intent Applications' (providing specific header, footer, and landing pages). A red arrow points from the browser to the text 'at browser, passing to one or more networks'. Below the browser is a 'Reverse Proxy', which routes traffic to three 'Shell (Middleware)' and 'LOB Middleware' components. The 'Shell (Middleware)' is owned by a central team, while the 'LOB Middleware' components orchestrate data calls to backends and send App/UI components to the browser. All three middleware components connect to '100s of RESTful Endpoints' at the bottom.</p> <p>Our front-end architecture matches pretty well to what Michael Geers might characterize as an “app shell” with multi-level routing. At the foundational level, we lean heavily on standard open source software libraries and practices. For backend libraries, we use (among others) Fastify, NestJS, Pino, Restify, and Undici. On the front-end, we utilize Vue.js and React.js.</p> <p>All of our browser to backend connections go through Node microservices. These microservices are responsible for most of the middle-tier work you would expect in a distributed application. This includes data orchestration, business rule enforcement, and logical decisioning.</p> <p>Source: https://www.capitalone.com/tech/software-engineering/loosely-coupled-micro-frontends-with-nodejs/</p> |
| receiving the one or more applets at the applet server manager from the at least one of the | The Accused Instrumentalities perform the step of receiving the one or more applets at the applet server manager from the at least one of the one or more networks. This is illustrated, for example, as follows: |

| | |
|--|---|
| <p>one or more networks;</p> |  <p>The diagram illustrates the front-end architecture. At the top, 'Capital One User' (who takes calls or helps solve problems) and 'Our Customers' (via 'Amazon Connect') interact with the 'Browser (We Provide The Shell, LOBs Provide Individual Web Applications)'. The browser layer contains a 'Shell' (managing routing, login, entitlements) and two 'LOB Business Intent Applications' (providing specific header, footer, content, and landing pages). This layer connects to a 'Reverse Proxy', which then routes traffic through 'Shell (Middleware)' (owned by a central team) and two 'LOB Middleware' components (orchestrating data calls to backends and sending App/UI components to the browser). Finally, all requests reach '100s of RESTful Endpoints'.</p> <p>Our front-end architecture matches pretty well to what Michael Geers might characterize as an “app shell” with multi-level routing. At the foundational level, we lean heavily on standard open source software libraries and practices. For backend libraries, we use (among others) Fastify, NestJS, Pino, Restify, and Undici. On the front-end, we utilize Vue.js and React.js.</p> <p>All of our browser to backend connections go through Node microservices. These microservices are responsible for most of the middle-tier work you would expect in a distributed application. This includes data orchestration, business rule enforcement, and logical decisioning.</p> <p>Source: https://www.capitalone.com/tech/software-engineering/loosely-coupled-micro-frontends-with-nodejs/</p> <p><i>at browser, passing to one or more networks</i></p> <p><i>server (backend)</i></p> |
| <p>processing the one or more applets at the applet server manager, wherein processing the one</p> | <p>The Accused Instrumentalities perform the step of processing the one or more applets at the applet server manager. This is illustrated, for example, as follows:</p> |

or more applets includes at least one of the following steps:



at browser,
passing to
one or more
networks

Our front-end architecture matches pretty well to what [Michael Geers](#) might characterize as an "app shell" with multi-level routing. At the foundational level, we lean heavily on standard open source software libraries and practices. For backend libraries, we use (among others) [Fastify](#), [NestJS](#), [Pino](#), [Restify](#) and [Undici](#). On the front-end, we utilize [Vue.js](#) and [React.js](#).

All of our browser to backend connections go through Node microservices. These microservices are responsible for most of the middle-tier work you would expect in a distributed application. This includes data orchestration, business rule enforcement, and logical decisioning.

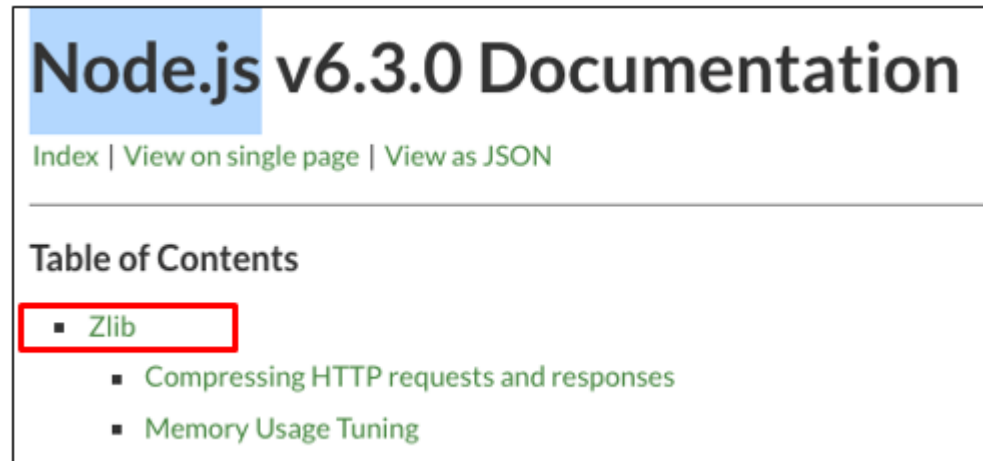
server (backend)

processing at backend with Node.js (Node)

Source: <https://www.capitalone.com/tech/software-engineering/loosely-coupled-micro-frontends-with-nodejs/>

compressing the one or more applets before sending the one or more applets to the one or more client computers, optimizing the one or more applets before sending the one or more applets to the one or more client computers, and verifying the one or more applets before sending the one or more applets to the one or more client computers; and

The Accused Instrumentalities at least perform the step of compressing the one or more applets before sending the one or more applets to the one or more client computers. This is illustrated, for example, as follows:



Node.js v6.3.0 Documentation
[Index](#) | [View on single page](#) | [View as JSON](#)

Table of Contents

- **Zlib**
 - [Compressing HTTP requests and responses](#)
 - [Memory Usage Tuning](#)

Source: <https://nodejs.org/dist/v6.3.0/docs/api/zlib.html>



Zlib **Compressing**

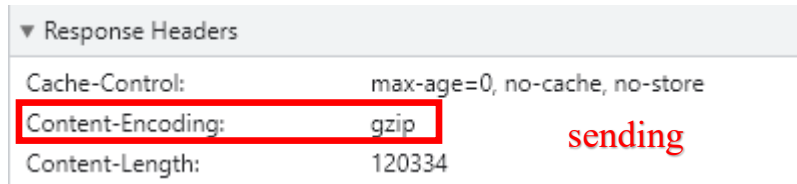
Stability: 2 - Stable

The `zlib` module provides compression functionality implemented using Gzip and Deflate/Inflate. It can be accessed using:

```
const zlib = require('zlib');
```

Compressing or decompressing a stream (such as a file) can be accomplished by piping the source stream data through a `zlib` stream:

```
const gzip = zlib.createGzip();
const fs = require('fs');
const inp = fs.createReadStream('input.txt');
const out = fs.createWriteStream('input.txt.gz');
```

| | |
|--|---|
| | <p>Source: https://nodejs.org/dist/v6.3.0/docs/api/zlib.html</p> <p>Upon information and belief, Node.js uses Zlib for compressing the one or more applets in one of the forms 'gzip' before sending it to one or more client computers.</p> |
| <p>sending the one or more applets from the applet server manager to the one or more client computers.</p> | <p>The Accused Instrumentalities perform the step of sending the one or more applets from the applet server manager to the one or more client computers. This is illustrated, for example, as follows:</p>  <p>Source: https://www.capitalone.com/credit-cards/fair-and-building/</p> <p>Upon information and belief, the applet is sent to the client computer in one of the forms 'gzip.'</p> |

Caveat: The notes and/or cited excerpts utilized herein are set forth for illustrative purposes only and are not meant to be limiting in any manner. For example, the notes and/or cited excerpts, may or may not be supplemented or substituted with different excerpt(s) of the relevant reference(s), as appropriate. Further, to the extent any error(s) and/or omission(s) exist herein, all rights are reserved to correct the same.